Lokalisierung von Objekten in Herbarbelegen

KLAUS PRÄTEL[1]

An der Fachhochschule Hannover wurde Mitte 2007 das Projekt "Herbar-Digital" gestartet. In dem Forschungsprojekt "Herbar-Digital" sollen aus 3,5 Millionen Papierbögen (Herbarbelege) des Botanischen Museums Berlin möglichst alle Objekte erkannt werden und separat verarbeitbar sein. Bei den Objekten handelt es sich um Barcodes, Tüten, Stempel, Farbtabellen, Elemente aus dem Pflanzenbereich sowie Hand- und Druckschriften. Es soll unter Zuhilfenahme des ADA-BOOST-Algorithmus vom Verfasser eine Objekterkennung realisiert werden, die folgende Eigenschaften aufweist: Position der zu erkennenden Objekte im Bild variabel, auch dreidimensionale - und konturschwache Objekte müssen erkannt werden, gleiche Objekte unterschiedlicher Form müssen erkennbar sein, das System muss lernfähig sein.

1 Einleitung

Im Projekt „Herbar-Digital“ wurde der Verfasser mit der Aufgabe betraut, Behälter, die Pflanzenteile enthalten, automatisch zu erkennen. Diese Behälter sind in Tütenform ausgeprägt. Die Problematik besteht darin, dass diese Tüten prinzipiell an jeder Stelle des Herbarbeleges in jeder Ausrichtung (geneigt oder gerade) anzutreffen sind. Außerdem ist durch die Füllung der Tüten, die erheblich sein kann, eine gewisse Unschärfe bei der Digitalisierung nicht zu vermeiden. Da die Herbarbelege beim Digitalisieren mit der Informationsseite nach oben auf dem Scanner angeordnet sind, ist es unvermeidbar, dass die Tütenverschlüsse eventuell leicht geöffnet sind. Dies alles erschwert die Erkennung.
Eine weitere Herausforderung sind stark unterschiedliche Tütenformen und Materialien. Bei den Materialien sind auch transparente Modelle vorhanden. Es ist leicht vorstellbar, dass diese besonders anspruchsvolle Erkennungsmethodiken erfordern.

2 Beschreibung der Vorgehensweise

Es muss eine Auswertungsstrategie erarbeitet werden, die Tüten erkennt und diese automatisch klassifiziert.
Es kann an der Tütenform, dem Verschluss und der Farbe eine Zuordnung (zumindest die Zeitzuordnung) zu dem Verfasser der Exponate getroffen werden.
Ein Problem ist auch die Anordnung der Tüte auf dem Exponat. In den meisten Fällen ist es schon so, dass die Tüte horizontal ausgerichtet ist. Dies muss jedoch nicht immer so ein. Ebenso ist der Ort der Tüte variabel.

1) Klaus Prätel, Fachhochschule Hannover, Ricklinger Stadtweg 120, 30459 Hannover;
E-Mail: klaus.praetel@fh-hannover.de

Vordergründig haben wir es mit 2 Problemen zu tun:

1. Ort und Ausrichtung der Tüte
2. Erkennung von Form, Verschluss und Farbe

Die Erkennung der Form der Tüte kann, um einigermaßen aussagekräftig zu sein, nur über Erkennungsalgorithmen erfolgen, die in Lernphasen zu immer mehr Informationen gelangt sind. Das Ergebnis der Lernphasen sind Klassifikatoren, die die möglichen Tüten, „repräsentieren“.
Aus diesem Grund werden Boosting-Verfahren untersucht.

Boosting ist ein genereller Ansatz verschiedene Klassifikatoren zu kombinieren, um eine verbesserte Gesamtperformance zu erzielen.

Es soll das ADAPTIVE BOOSTING als ein Vertreter der Boosting-Verfahren betrachtet werden.

Die wesentliche Idee bei allen Boosting-Algorithmen ist, statt eines starken Klassifikators, schwächere Klassifikatoren für ein Entscheidungsproblem einzusetzen, indem man ihre Ergebnisse kombiniert. Der Gedanke dabei ist, dass wesentlich leichter ein paar Faustregeln für ein Problem zu finden sind als eine generelle Regel, die das Problem löst.
Angenommen, es soll ein Programm entwickelt werden, das beim Pferderennen auf den Gewinner tippen soll. Gegeben sind übliche Informationen, wie zum Beispiel die Anzahl der bereits gewonnenen Rennen bei jedem Pferd. Um eine Entscheidungsregel für das Programm aufzustellen, wird ein Experte nach seiner Meinung gefragt. Es ist offensichtlich, dass es ihm schwer fallen wird, eine einzige Regel aufzustellen, anhand derer man auf den Gewinner schließen kann. Umgekehrt ist es aber für ihn viel leichter, ein paar Faustregeln zu nennen (wie zum Beispiel “Tipp auf das Pferd, das in der letzen Zeit oft gewonnen hat” oder “Tipp auf das Pferd, das die besten Quoten hat ”). Es ist einsichtig, dass solche Regeln allein nur ein bisschen besser sind, als zufällig auf ein Pferd zu tippen. Hat man andererseits viele solche Daumenregeln aufgestellt und kombiniert sie zu einer gewichteten Mehrheitsentscheidung, so erhält man einen mächtigen Entscheidungsalgorithmus.
Die Hoffnung besteht darin, dass so kombinierte Daumenregeln bessere Ergebnisse liefern werden, als eine allgemeine Regel. Boosting ist ein Verfahren, das eine effiziente Entscheidungsregel für ein Klassifikationsproblem aufstellt, indem es mehrere einfache Regeln kombiniert. Diese Regeln werden im Folgenden schwache Klassifikatoren oder Basisklassifikatoren genannt. Das Ergebnis dagegen, also die akurate Entscheidungsregel, bezeichnen wir als starken Klassifikator.

Die Idee, Boosting Verfahren für Klassifikationsprobleme einzusetzen, ist relativ neu.

Die ersten Veröffentlichungen zu dem Thema sind 1990 von Freund erschienen. Fünf Jahre später ist der AdaBoost-Algorithmus vorgestellt worden. AdaBoost hat erstaunlich gute Ergebnisse für sämtliche Klassifikationsprobleme geliefert und hat sich somit in den letzten zehn Jahren eindeutig durchgesetzt. Deswegen steht auch AdaBoost und seine Varianten im Fokus

dieser Ausarbeitung.

Boosting ist eine effektive Methode, einen starken Klassifikator zu erstellen, indem man schwache Klassifikatoren kombiniert. Es ist relativ einfach zu entwerfen und hat fast keine Parameter einzustellen. Die einzige Designfrage ist, wieviele und welche Weak-Learner (schwache Klassifikatoren oder Basisklassifikatoren) verwendet man zum Trainieren. In der Praxis aber geht man meistens so vor, dass man die Weak-Learner für das gegebene Problem bereits gewählt hat. Und dann bietet Boosting ein Gerüst, um sie zu kombinieren. Dadurch, dass der Algorithmus sehr allgemein ist, kann Boosting mit jeder Art von Weak-Learnern umgehen. Dabei reicht es aus, dass die Weak-Learner Hypothesen ausgeben, die nur etwas besser sind, als eine zufällige Hypothese. Sind also die Weak-Learner gut gewählt worden, so erhält man eine Garantie, dass sowohl der Trainings- als auch der Generalisierungsfehler exponentiell sinken. Das in der Ausarbeitung vorgestellte AdaBoost ist der Vertreter der Boostingverfahren und hat sich in den letzen Jahren eindeutig durchgesetzt. AdaBoost und seine Varianten bieten Lösungen nicht nur für Zwei-, sondern auch für Mehrklassenprobleme. Die Experimente mit AdaBoost zeigen, dass AdaBoost sehr oft genauso gute und manchmal sogar viel bessere Ergebnisse liefert als andere Entscheidungsverfahren.

2.1 Theoretische Betrachtung

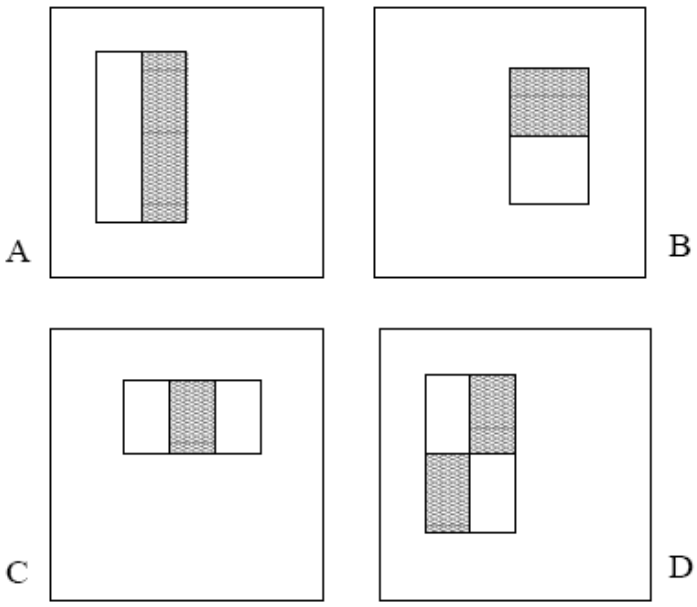

Abb. 1: Beispiel-Rechteck-Features

Die in Abb. 1 gezeigten Features werden dargestellt in ihrer relativen Position zum zu untersuchenden Fenster. Die Summe der Pixel, die in dem weißen Rechteck liegen, werden von der Summe der Pixel, die in dem grauen Rechteck liegen, abgezogen. Zwei-Rechteck-Features werden in der Abbildung 1A und 1B gezeigt, ein Drei-Rechteck-Feature in Abbildung 1C und ein Vier-Rechteck-Feature in Abbildung 1D.

Features

Object-Detection-Procedures klassifizieren Images, die auf einer Anzahl von einfachen Features basieren. Es gibt viele Gründe, warum eher Features benutzt werden als direkt die Pixel zu verarbeiten. Ein sehr wichtiger Grund ist, dass Feature-basierte Systeme erheblich schneller sind als Pixel-basierte Systeme. Die einfachen Features erinnern an die Funktion, die auf dem Haar-Prinzip basieren. Nehmen wir an, die Auflösung des Detektors ist 24*24 Pixel, dann ist der vollständige Umfang der Features sehr groß, nämlich 45396. Eine komplette Basis hat keine lineare Abhängigkeit zwischen den Basiselementen und hat die gleiche Anzahl an Elementen wie der Bildbereich, in unserem Fall 576. Der volle Satz von 45396 Features ist etliche Male, wie man sagt, over complete!

Integral Image

Rechteck-Features können schnell verarbeitet werden, wenn man über eine Zwischenbild-Darstellung geht, die Integral Image genannt wird. Das Integral Image (siehe Abb. 2) an der Stelle x,y beinhaltet die Summe der Pixel oberhalb und links von x,y:

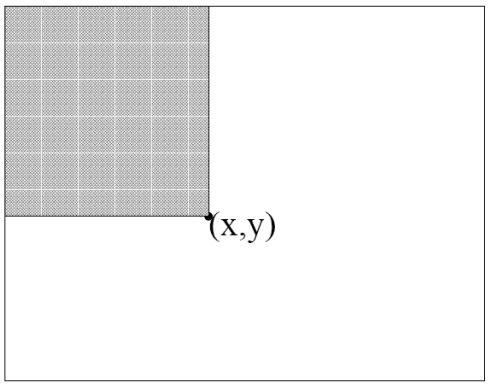

Abb. 2: Integral-Image

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y')$$

wobei ii(x,y) das Integral Image ist und i(x,y) das Original Image. Benutzt wird das folgende Paar von Rekursionen:

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

(wobei s(x,y) die kumulierte Reihensumme ist, s(x,-1)=0, und ii(-1,y)=0), das Integral Image kann in einem Durchlauf über das Original Image gebildet werden.
Benutzt man das Integral Image, kann jede Rechtecksumme über 4 Feldreferenzen (siehe Abb. 3) bearbeitet werden.

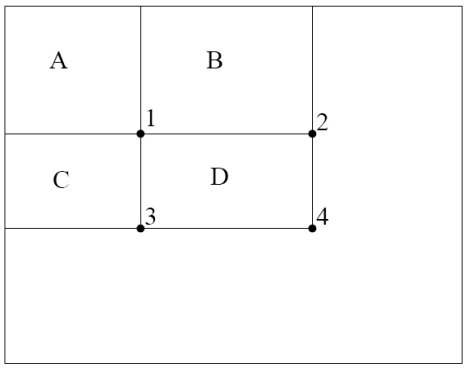

Abb. 3: Integral-Image Feldreferenzen

Die Summe der Pixel im Rechteck D wird berechnet aus 4 Feldreferenzen. Der Wert des Integral Image an der Stelle 1 ist die Summe der Pixel im Rechteck A. Der Wert an der Stelle 2 ist A + B, an der Stelle 3 A + C und an der Stelle 4 A + B + C + D.
Die Summe in D wird berechnet 4 + 1 – (2 + 3).

Klassifikationsfunktion

Zur Erinnerung: Jedes Image Sub-Window hat 45396 Rechteck-Features, eine Anzahl, die erheblich größer ist als die Anzahl der Pixel. Wenngleich jedes Feature sehr effizient bearbeitet werden könnte, ist die Bearbeitung aller Features überaus aufwändig. Die Hypothese ist, dass eine kleine Anzahl von Features zu einem effektiven Klassifikator vereint werden kann. Die Hauptaufgabe ist es, diese Features zu finden.

In dem verwendeten System wurde eine Variante von AdaBoost benutzt, die zum Selektieren der Features und zum Trainieren des Klassifikators geeignet ist. In der Originalform wird der AdaBoost-Lernalgorithmus zum Verbessern der Klassifikations-Performance eines einfachen

Lernalgorithmus benutzt. Dies funktioniert über Verbinden eines Pools von Weak-Klassifikatoren-Funktionen, um einen starken Klassifikator zu erzeugen. In der Sprache des Boosting ist ein einfacher Lernalgorithmus ein Weak-Learner. So sucht z.B. der Perceptron-Lernalgorithmus über einen Satz von möglichen Perceptren und liefert ein Perceptron mit dem kleinsten Klassifikations-Fehler. Der „Lerner“ wird „weak“ genannt, weil man nicht die bestmögliche Klassifikationsfunktion zum Klassifizieren von Trainingsdaten erwartet (für ein gegebenes Problem reicht ein Perceptron aus, das die Trainingsdaten korrekt zu 51% klassifiziert) . Um einen Weak-Learner zu boosten, gilt es einige Lernprobleme zu lösen. Nach der ersten Runde des Lernens müssen die Trainingsdaten derart gewichtet werden, dass die, die unkorrekt klassifiziert wurden, durch den vorigen weak-Klassifizierer in ihrer Wichtung angehoben werden.
Der abschließende starke Klassifikator nimmt die Form eines Perceptrons ein.

Die konventionelle AdaBoost-Prozedur kann als ein „gieriger“ Feature-Auswahl-Prozess bezeichnet werden.
Abschließend kann gesagt werden, dass AdaBoost eine effektive Prozedur zum Suchen von einer kleinen Anzahl von guten Features ist, die eine signifikante Vielseitigkeit haben.

2.2 Praktische Anwendung

Zur Anwendung kam ein Softwarepaket der Firma Intel, genannt openCV. Es müssen insgesamt 4 Schritte durchlaufen werden:

1. Vorbereitung
2. Sample-Erzeugung
3. Training
4. Test
5. Bei Bedarf ist ein Performancetest der Kaskade vorzunehmen.

2.2.1 Vorbereitung

Es müssen zum Training positive (Objekt im Bild vorhanden) sowie negative (Objekt im Bild nicht vorhanden) Samples vorhanden sein. Wegen der Kompressionsartefakte in einigen Kompressionsleveln von JPEG-Images ist ein BMP-Imageformat zu bevorzugen. Die Anzahl der negativen Samples sollte sich in der Größenordnung von 5000 bis 10.000 Bildern bewegen. Sicher ist eine Anzahl der positiven Samples in der gleichen Größe wünschenswert, jedoch ergeben auch erheblich kleinere Mengen ein akzeptables Ergebnis (500 bis 800).

2.2.2 Sample-Erzeugung

Es wird davon ausgegangen, dass eine Samplegröße von 20x20 eine gute Größe für die meisten Objekte ist. Deshalb werden die Objekte auf diese Größe reduziert.
Grundsätzlich werden 4 Sätze von Bildern bearbeitet:

1. Ein positiver Exemplarsatz, der d das Objekt darstellt, auf das trainiert werden soll
2. ein anderer positiver Bildersatz für Testzwecke
3. ein negativer Bildersatz (ein sogenannter Hintergrund) zum Trainieren
4. ein zusätzlicher negativer Bildersatz zum Testen

Achtung: Die Testsätze sollten keine images enthalten, die für das Training benutzt werden!!

Natürlich muss die Verteilung der Images in Training und Test sinnvoll angelegt sein. Die Erfahrungen haben gezeigt, dass bei z.B. 5500 images die Verteilung 5350 für das Training und 150 images für das Testen angebracht sind.

2.2.3 Training

Zum Training der Kaskaden werden dem Trainingstool die Sammlungen der positiven sowie negativen Samples zur Verfügung gestellt. Das Ergebnis ist eine Kaskade, die vom Testingtool benötigt wird.

2.2.4 Testing

Das Testingtool bekommt als Eingabeparameter die Kaskade des Trainings sowie die zu untersuchenden Samples. Bei Erkennung des Objektes wird dieses durch ein grünes Rechteck gekennzeichnet. Der zu testende Herbarbeleg wird dem Testtool auf direktem Wege, also über eine Web_Cam, zur Verfügung gestellt oder es wird ein Herbarbeleg in Form eines Files des Formates .BMP angeboten.

3 Ergebnisse

Am Beispiel in Abb. 4 ist zu erkennen, dass die Art des Tütenverschlusses stark variieren kann. Man sieht am oberen Objekt, dass der Verschluss leicht geöffnet ist und somit etwas unscharf dargestellt wird. Dies beeinträchtigt jedoch die Erkennung nicht. Das untere Objekt zeigt keinen sichtbaren Verschluss, auch dies führt nicht zum Fehler.
In Abb. 5 wird dem Testprogramm der Herbarbeleg über eine Web-Cam zur Erkennung des Objektes zur Verfügung gestellt. Im mittleren Bildausschnitt ist das grüne Rechteck zu erkennen, das das Objekt umringt. Da es sich in dieser Anwendung um eine Life-Auswertung handelt, folgt das grüne Rechteck dem Objekt, wenn der Herbarbeleg vor der Kamera bewegt wird.
Dies wird bei dem Vortrag des Verfassers vorgeführt.

Abb. 4: 2-Fach Objekterkennung BMP

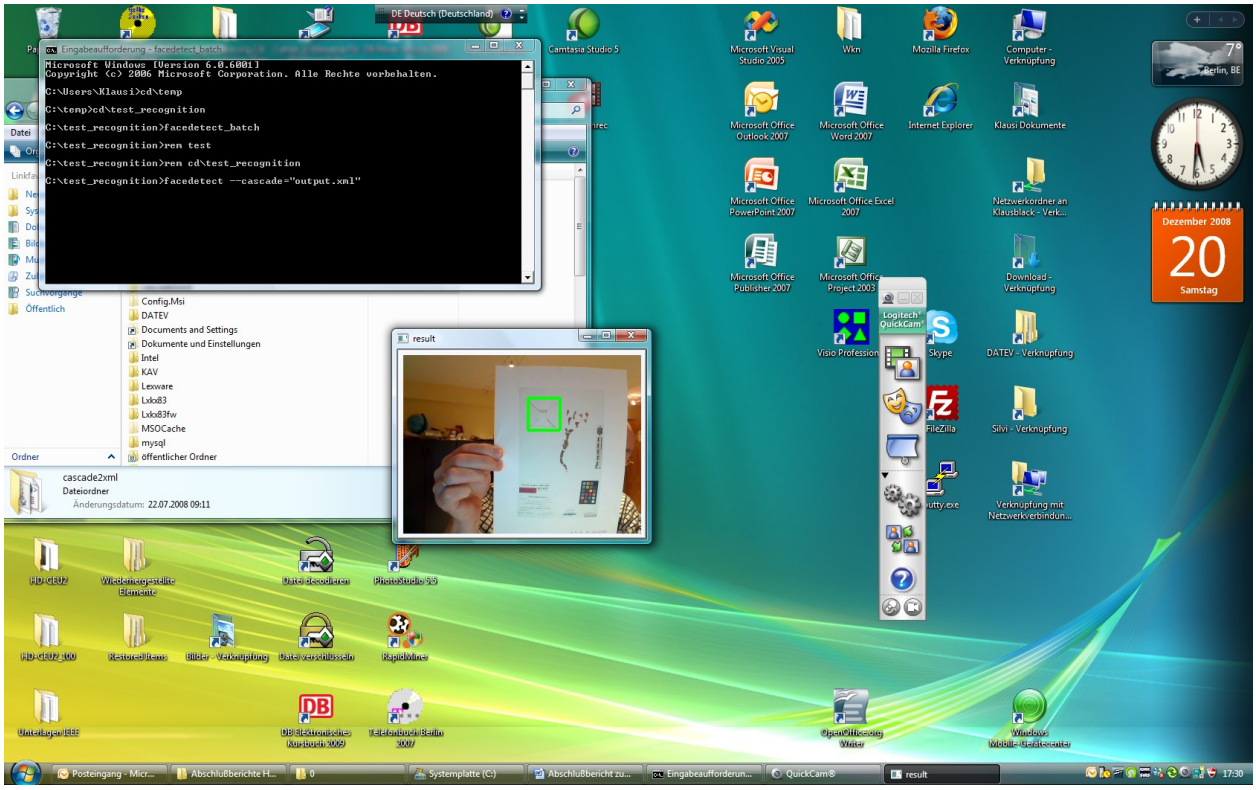

Abb. 5: Objekterkennung über Web-Cam

4 Auswertung

Es wurden insgesamt 549 Herbarbelege ausgewertet. Durch Vorkommen von mehreren Objekten pro Herbarbeleg ergeben sich in den Summen der zu verarbeitenden Objekte größere Werte als die Anzahl der Herbarbelege. Besondere Zustände der Objekte sind explizit in der Auswertung beschrieben.

Abb. 6 zeigt die Aufsummierung der Ergebnisse für HIT (Treffer) , MISS (nicht erkannt) und FALSE, (falsch erkannt) .

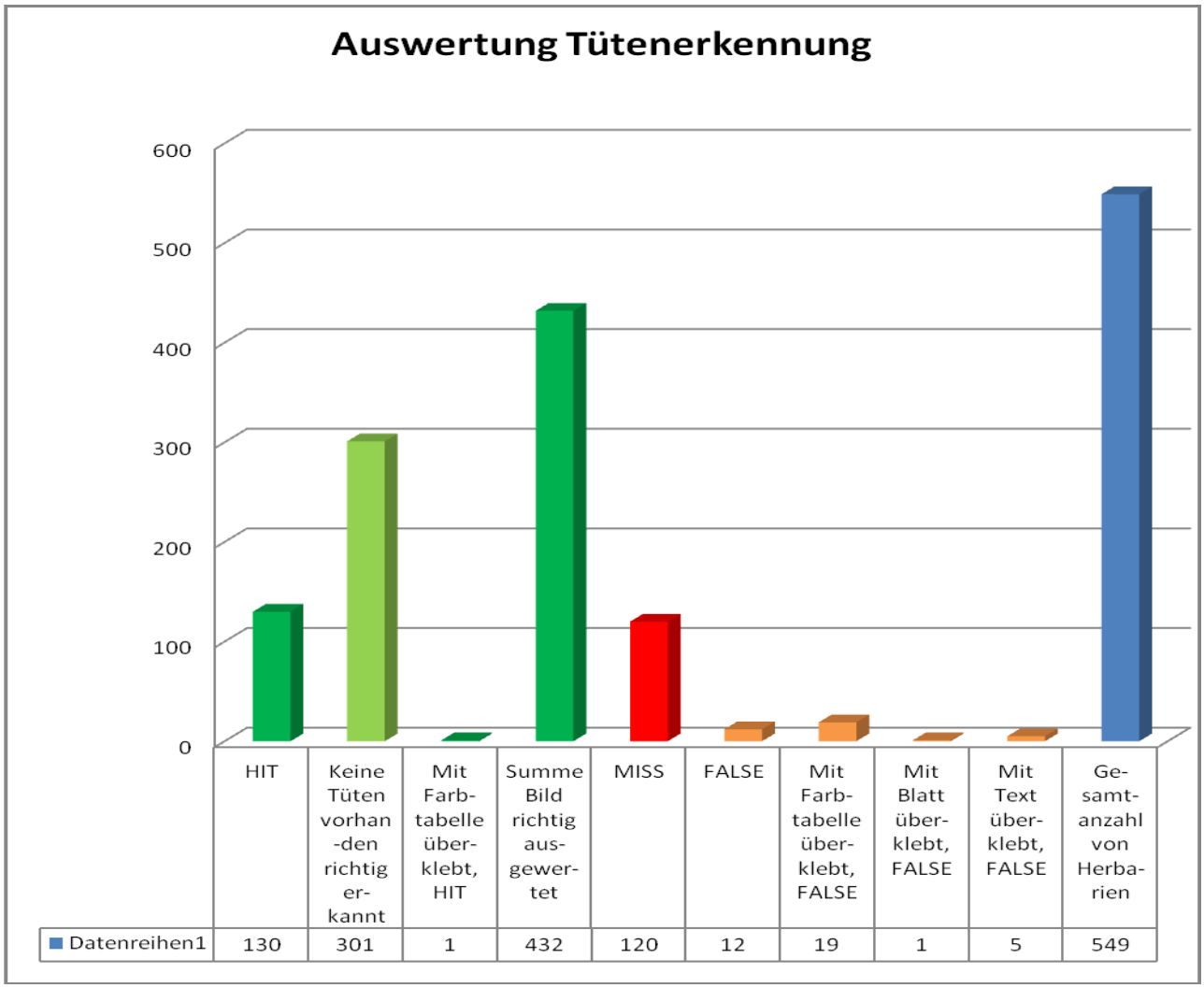

Abb. 6: Auswertung Tütenerkennung

Auszug aus Performance-Test der Kaskade. Dieser zeigt das Ergebnis aus Abb. 7.

3 HIT, 0 MISS, 2 FALSE.

Bezeichnung	HIT	MISS	FALSE
4_Herbarium230.bmp	3	0	2
4_Herbarium231.bmp	1	0	2

Abb. 7: 4_Herbarium230.bmp

5 Literaturverzeichnis

Viola, P.,Jones, M., Robust Real-time Object Detection, Vancouver, Canada, July 13, 2001

Adolf, F., How to build a cascade of boosted classifiers based on Haar-like features, September 2, 2003

Steinke, K.-H., Dzido, R., Gehrke, M., Prätel, K., Feature recognition for herbarium specimens (Herbar-Digital), Proceedings of TDWG, Perth, 2008